RESUME OF A 4-YEAR-OLD,
47,250-POUND SOFTWARE EXPERT

informatics inc®

Design and Development of Software Packages

by RICHARD H. HILL
Vice President/Programming
Informatics Inc.

Reprinted with permission from American Management Association, Management Bulletin 79, published and copyrighted 1966.

informatics inc.

Main Offices:

5430 VAN NUYS BOULEVARD
SHERMAN OAKS, CALIFORNIA 91401

4720 MONTGOMERY LANE
BETHESDA, MARYLAND 20014

DESIGN AND DEVELOPMENT OF SOFTWARE PACKAGES

by RICHARD H. HILL
Vice President/Programming
Informatics Inc.

THE SOFTWARE INDUSTRY IS A YOUNG ONE, comparatively speaking. It is hard to know just where to date its beginning, but seven or eight years ago is reasonably close. Six or seven years ago the total market for the services of independent software companies was somewhere on the order of $5 million a year. Currently, Informatics Inc. alone does business at that annual rate, and the total market for the software industry is probably somewhere between $75 million and $100 million a year. This estimate excludes the computer time that is sold by companies which are also in the software business. These figures are my credentials, as a representative of a software company, for discussing the design and development of software packages.

RICHARD H. HILL is Vice President—Programming, West Coast Office, Informatics Inc., Sherman Oaks, California.

At Informatics a software package is defined as "an integrated set of routines and associated documentation to accomplish practically anything that can be accomplished through EDP." Most of the work done by Informatics is of a type that can't be done through procedure-oriented or even problem-oriented languages. Our work is done usually in the language of the machine as represented by symbolic assembly programs, because we are usually called upon to implement systems that are a little bit out of the ordinary. Our specialty area is the field of on-line, real-time, man-machine systems, which are relatively complex. We design and build executive routines and monitor systems, display-programming systems, message-switching systems, and the like. Thus we think of software in terms of systems programs—that is, the programs that support the development of applications packages. However, I don't want to exclude any type of software package from consideration when discussing the problems of design, development, and—particularly—management of software activities. I think that the management problem especially is one that exists on all levels of planning and developing programs.

The basic concepts that were important in the development of modern computing systems are three. The first was the concept of symbolic language; it introduced the notion that one could express computer programs in a language other than that of the computer. This concept was first implemented by symbolic assembly programs, then by procedure-oriented languages, and today by some problem-oriented languages. The second basic concept was that of the generalized routine—a routine once written can be used over and over again as part of many software packages. Generalized routines were first used to do such things as work out square roots or sines and cosines. There are now a great number of generalized routines that do everything from those mathematical functions to sorting, report generation, and so on. The third important concept in the development of software packages was the notion of automatic transitioning from one activity to another, coupled with the idea of letting the computer do some of its own accounting work. From these basic concepts has grown the whole family of systems programming activities, including compilers, monitor systems, executive routines, generalized input-output systems, and the like. From a management viewpoint the batch-process monitor, in which jobs are introduced to the computing system one after another in a continuous stream, was a most significant forward step. The computer provides the transition from one activity to the next and does the necessary accounting as it goes. However, this is essentially a serial process in which the efficiency of the total computing system is limited by the interaction of the program with its data.

As the relative cost—that is, the cost-effectiveness—of hardware improved it became possible to utilize large segments of hardware for overhead functions and thus make feasible the timesharing of the equipment through multiprogramming. This step made it possible for jobs to be mixed and for the efficiencies of one job to be used to offset the inefficiencies of another. In the ideal situation a computer-bound program shares the computer with an input-output-bound program, and both of the programs together make maximum use of the available hardware.

The feasibility of these approaches really was not established until the current generation of computers, when electronics became so cheap and so reliable as to make feasible the dedication of a significant amount of hardware to the overhead functions necessary to allow such things as multiprogramming to take place.

Now there is a concomitant that must be accepted along with the great capability which is present in the third generation of computers. That concomitant is that all these overhead functions are accomplished by programs and, furthermore, these programs are among the most complex yet encountered. An example of this complexity is a program which has some 10,000 instructions in it, about 80 percent of which are yes-no decisions. The number of paths through that program is literally incalculable. This particular program is a systems program that is part of a large, modern computing system. It is something of a paradox that the hardware for this system is inexpensive and reliable—probably, the hardware is more reliable than the software. But we don't really have any good way of testing the software other than exercising it for a considerable length of time. In this respect, as in

many others, the hardware technology in computing is far ahead of software technology.

This problem of complexity in software to a great extent becomes the management problem of knowing where to cut off the development process. Cutting off the specification process before implementable limits are reached results in control. Anything less is tantamount to allowing invention to run wild.

This concept can be developed a little further. We have seen the use of the term "architecture" in connection with computer systems today. It is an apt term because it refers to the fact that software has developed beyond the point where the programmer is an artist. In effect, he has become an architect. The distinction is this: In the infancy of computing, a man approached a bare computer with an understanding of how the hardware worked and essentially created through his own artistry a program to solve a given problem. He may or may not have left traces other than the code behind him. His artistry was revealed in the code itself, and if it ran it was a wondrous thing. But the modern computing architect cannot afford this luxurious kind of creation. Rather, he has to view his activity as part of a complex, interacting system, just as a large, modern building is a complex, interacting system. The scope of his design activities has grown such that, although one person can possibly conceive the grand design for a large programming system, he cannot possibly hope to carry it out by himself down to the finest detail of design. So, like the architect who designs large, modern buildings, the architect of computing systems must communicate his grand design to his associates and coordinate their activities in its implementation.

This is easy to say, but it is a very difficult thing to do, because this sort of architecture is really in its infancy. Anybody who has been associated with the development of IBM System/360 would agree that the concept of architecture certainly exists there. The implementation has been a matter of infinite pain to a great many people. This is not to say that the implementation will not be good or that it will not be achieved, but the fact that programmers lived through something like that attests to the tenacity of the human mind and its inexhaustible appetite for challenge.

There are many management problems associated with the successful implementation of a large-scale system. These problems are magnified by the fact that the art of programming really has not progressed to the point where it is able to cope with large-scale efforts on a readily achievable scale. The basic techniques—the tools of our profession—have not been developed to the point where we are able to say with certainty how much, for example, a given program will cost or how long it will take to write it, even after it has been described to us in great detail. Consequently, in our particular business there is a high element of risk involved in many types of activities. But it is the type of risk that we in the software industry are willing to accept on the basis of having had a considerable degree of experience, at least at a certain level. However, for people who do not have a similar reservoir of experience there are no tried and true methods to substitute for the experience. For example, there is no standard way of acquiring, or even stating the requirements of, a program in a sufficiently general form to be able to provide a general solution. We are reduced to the technique of analysis which consists of asking people, "What is it you want?" This question may have to be asked more than once because very frequently people don't know what they want or don't know how to state it. It is not infrequent for people to make changes as the program develops and even after it is completed. This is normal; this is why program maintainers exist. But it does point out the problem of finding a language which is sufficiently general to state requirements and the problem of finding programming techniques that are sufficiently flexible to permit the inevitable changes to be made at minimum cost and minimum risk of introducing new errors.

At Informatics Inc. we identify in the programming process some six or seven basic functions. First, when we attack a problem we prepare what we call a functional specification, which is the initial statement of requirements that we give to a customer and ask that he approve as the basic working document for further

design activities. Once we have agreed on a functional specification, which is effectively a statement of requirements, we produce a program design specification in which not only the externals of a program but also the internal make-up of the program, its various constraints, its reactions with other programs, and so forth are described. When this document is completed to our satisfaction and the customer's, we enter a coding phase. During this coding and the subsequent check-out of the program, we will be doing final documentation. And usually after check-out there will be some additional documentation to complete. Finally we will install the program—that is, bring it to a customer's location, make sure it runs on his computer, and train his people in its use and maintenance.

Note that only two of these six steps are directly concerned with the machine—the coding and check-out phases. The others are concerned with thinking and paperwork. If there were a less traumatic way to prepare documentation than by writing it down we probably would use it. But the fact remains that, in terms of today's techniques, the best way we have of capturing the ideas that go into a program and describing what a program does is by writing it down.

Documentation is all important because we don't document just for ourselves but primarily for other people. And the impression that we make has to be a favorable, lasting one or else we don't get any repeat business. Documentation thus is an absolutely essential investment from a management point of view because, unless the documentation comes with the program, the money spent on the program is in effect wasted, whether it was an in-house development or an outside purchase.

The first law of programming management is not to let programmers make decisions—at least, not until they become something more than programmers. All too often in the past, decisions have been made in the computing shop by default.

To leave important decisions, such as what your programming languages should be, to a person who has naturally a biased interest in the decision—the programmer himself—is to defeat the effectiveness of the program as well as of the program manager. Programmers don't usually think like managers. They think like people, people who want to take the least painful route to accomplishing a set task and get it over with as soon as possible. These goals unfortunately are inconsonant with management's goals. The first problem of a manager of programmers, no matter what level he is operating on, is to understand that he has to impose some degree of management discipline upon his programming activities or else the whole thing is going to fail.

Programming is a manageable activity, but it takes a high degree of skill and knowledge of programming in order to manage it properly. Programming activities should be managed much like any other engineering or development activity, and it is very important to make preliminary estimates and budgets and to develop detailed schedules. In fact, Informatics uses the PERT technique when its projects are big enough. There should be a rigid and continuing system of monitoring the programming process, which in turn implies that functions in programming situations must be monitorable. In other words, you must establish milestones so that a check can be made to see whether or not the milestones have been met.

It is evident, then, that the important thing for management to understand about software development is the programming process. Reasonable schedules and reasonable milestones cannot be established unless management understands what the programmer has to go through to accomplish his work. One of the problems encountered repeatedly in selling software to private industry is lack of realism in looking at the software development process. This is found less often in selling software to government, where military and NASA have provided a pretty large background experience. But all too frequently, people in private industry expect the specification model to be absolutely correct. Of course, it isn't. Neither are the programs the first time around.

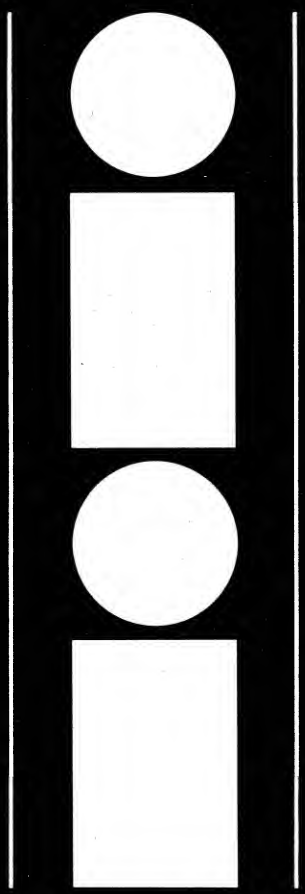
In addition, management must realistically assess the capabilities of its staff—in other words, be sure what its programmers can do. Of the

some 100,000 people in the United States today who are writing instructions for machines, there are perhaps only about 10,000 who are really professional programmers. Finally, sound ways to measure programming productivity must be developed. One of the reasons the whole profession has fallen behind in establishing measures is that the types of programming activities haven't been defined well enough so that we can measure productivity. Measures such as two instructions per hour, 40 instructions per day, and 60 instructions per day are realistic for certain types of programming activity. Yet more complex jobs may be estimated at one instruction per hour, ten instructions per day—and even these estimates may not be conservative enough. The difference, of course, is in the relative complexity of the jobs undertaken.

The amount of documentation also varies from job to job. Some programs may demand one page of documentation for every 40 instructions in the finished program, in addition to the comments and the listings. Thus, for a system we recently produced that had about 45,000 instructions, there were 1,000 pages of documentation. The documentation included program descriptions, flow charts, write-ups, functional specifications, and so forth. This is a considerably higher level of documentation than most programs produce or even need; for example, a sort run on a well-defined master file certainly doesn't require a high level of documentation, even though there may be far more instructions involved.

When practical measures are established and related to problem complexity, management will be able to use these tools to establish more effective control of software. Efforts of independent software organizations and talented individuals should be directed toward this goal.

informatics inc./CAPABILITIES



Who We Are □ Informatics Inc. is an independent software firm specializing in the design, analysis, programming and implementation of computer based systems for Government and Industry.

Informatics was established in 1962. Since that time we have provided our services to over 100 well-known companies and Government agencies.

The company has major offices on the East and West coasts of the United States, has operating units in six U.S. cities, and has a field office in Europe.

What We Do □ Informatics performs a variety of tasks in data processing — from design consulting to total implementation of large scale systems. Informatics usually assumes total systems implementation responsibility, including services such as: system analysis, system engineering, program design, implementation, documentation and system operation. In some cases professional assistance is provided the customer in view of personnel shortages or to fill a requirement for specialized experience.

The company also designs and develops, for sale or lease, proprietary data processing programs for a broad spectrum of tasks.

Our Management Philosophy □ Informatics' basic approach is to effect a strong discipline of project planning and monitoring. The Project Manager is required to prepare a project plan covering all technical and administrative matters. This is required for all projects — large and small. The Project Plan is updated monthly and is reviewed monthly by company management.

A well-run company is one which has good customer relations and is profitable. Informatics scores exceptionally high on both counts. We believe our record for customer satisfaction is unmatched. A company which manages its own financial affairs is likely to handle customer projects well also. Informatics has had a record of continuously profitable operations since early 1963.

OUR PROFESSIONAL CREDENTIALS

The Informatics staff has published over 40 papers on data processing in major publications. These have covered the most modern topics – management and technical – in the field today. Informatics personnel are frequent speakers and participants in professional meetings of the Association of Computing Machinery and the American Federation of Information Processing Societies, and have held many national offices in those organizations.

The company has sponsored four national symposia on modern topics:

Disc File Applications, 1964
On-Line Systems, 1965
Computer <=> Graphics, 1966
Computers and Communications — Toward a Computer Utility, 1967

The latter three were sponsored in conjunction with the University of California at Los Angeles. In each case speakers of national renown participated, and hard cover proceedings of each are available through publishers. Also, in each symposium, Informatics personnel made technical presentations.

OUR STAFF

From the best of many outstanding scientists, Informatics has built a highly competent technical staff – a staff which combines the best qualities of experience, thoroughness, technical imagination, and the skill to communicate.

Informatics people are experienced – with the machines and operating systems of nearly every manufacturer, and with a broad spectrum of applications.

The forty senior staff members of Informatics average 15 years' experience with modern information systems – a field which is less than 17 years old. Our staff members have direct working knowledge of over 250 programming languages, including such procedure-oriented languages as FORTRAN, COBOL, ALGOL, NELIAC, JOVIAL, PL/I and 120 operating systems languages.

ON-LINE SYSTEMS OUR SPECIALTY

Within its broad spectrum of experience, Informatics has a specialty: on-line or real-time systems.

Informatics has been a leader in the development of on-line systems – those systems in which central processing units must respond to demands of such external equipment as displays, communication devices, interrogation stations, and instrumentation. Approximately 80% of Informatics' work is with on-line systems, which we believe to be the most advanced and challenging computer systems.

Informatics has specific experience in on-line systems of many types: missile ground systems, military command and control, intelligence data handling, on-line displays, commercial communication systems, large scale time sharing systems, and commercial reservation systems. In addition, Informatics has designed and implemented operational programs for several major computer/communication networks; these include the GSA network for all U.S. Government civilian agencies, Western Union's Telex system, and NASA's global NASCOM network.

ADDITIONAL AREAS OF ACTIVITY

Advanced Information Systems □ Design and development of large-scale management system programming packages – general purpose File Management Systems, Management Information Systems, and industry-wide commercial computer applications. We have provided metropolitan data banks for several cities throughout the United States. Four other sophisticated general purpose file management systems are now in operation.

Command and Control □ The design and implementation of communications based systems involving on-line displays and large scale files. Informatics has performed on such systems as: National Military Command System, the Army Operations Center, and the Navy's FLAGPLOT, each of which is the top-level command and control system for the JCS, the Army and the Navy, respectively.

System Engineering □ Extending Informatics' software capabilities in the fields of hardware/software interface analysis and design, computer-based communication system design, and system requirements determination and integration has been applied to major command and control system designs for the military. We have provided a computer design concept for Navy shipboard command and control and for a Marine transportable land based command and control system, as well as system engineering assistance on several large-scale military projects.

CPM Systems □ Application of network planning, critical path analysis, and related resource allocation methods of production problems, particularly in the construction industry. We have provided computer based construction scheduling for over 9,000 units and CPM management assistance for large-scale military and commercial projects.

We have successfully completed and are currently engaged in performing programming contracts in the following major areas:

Systems	Executives Assemblers Compilers Sorts Data Management Utility Software Language Processors Graphics Diagnostics
Scientific	Aerospace Engineering Design Automation and Numerical Control Numerical Analysis Test Data Reduction Engineering Information Sciences Synthetic Intelligence Stochastic Simulation
Commercial	General Accounting Retail Trade Banking New Product Analysis Management Information Systems Manufacturing Insurance Accounting Advertising Accounting Credit Systems
Systems Applications	Critical Path Methods Real-Time Data Handling Message Switching Displays Range Systems Intelligence Communications Systems Multi-computer Systems Command and Control General Purpose File Management

WE ARE PROUD OF OUR CLIENTS

Our clients include major corporations and Government agencies engaged in advanced electronic systems and significant information processing efforts. **Among them are:**

Air Force Electronics Systems Division
Army Personnel Research Office
Bendix Corporation
Bunker-Ramo Corporation
Carson/Roberts
City of Alexandria, Virginia
Control Data Corporation
Department of Health, Education and Welfare
General Dynamics/Electronics
General Electric Corporation
General Instrument Corporation
Honeywell EDP
International Business Machines
Jet Propulsion Laboratory
Litton Industries
Music Corporation of America
National Aeronautics and Space Administration
National Cash Register Company
National Military Command Support Center
National Security Agency
North American Aviation
Office of Naval Research
U.S. Navy Missile Test Center, Pt. Mugu
Paramount Pictures
Perkin-Elmer Corporation
Philco/Ford
Philips Gloeilampenfabrieken
Radio Corporation of America
Raytheon Computer Company
Rome Air Development Center
Scientific Data Systems
Shell Oil Company
State of New York
TRW, Inc.
United California Bank
U.S. Department of State
UNIVAC
Western Union

informatics inc.

Corporate Offices

5430 Van Nuys Boulevard
Sherman Oaks, California 91401
Phone: (213) 783-7500

European Office
Building 64
Schiphol Airport
Amsterdam, The Netherlands
Phone: 156-297

Western Operations

5430 Van Nuys Boulevard
Sherman Oaks, California 91401
Phone: (213) 783-7500

16811 El Camino Real
Houston, Texas 77058
Phone: (713) HU 8-1627

2000 Riverside Drive
Los Angeles, California 90039
Phone: (213) 663-8391

Eastern Operations

4720 Montgomery Lane
Bethesda, Maryland 20014
Phone: (301) 654-9190

467 Sylvan Avenue
Englewood Cliffs, New Jersey 07632
Phone: (201) 568-6420

Henry Building
Upper Turin Road
Rome, New York 13440
Phone: (315) 337-5600

10 Mount Auburn Street
Watertown, Massachusetts 02172
Phone: (617) 924-3560

NAME: Informatics Inc.

ADDRESS:

Headquarters:
5430 Van Nuys Boulevard
Sherman Oaks, California
(213) 783-7500

Eastern Operations:
4720 Montgomery Lane
Bethesda, Maryland
(301) 654-9190

Other Locations:

467 Sylvan Avenue
Englewood Cliffs, New Jersey
(201) 568-6420

16511 El Camino Real
Houston, Texas
(713) HU 8-1627

2000 Riverside Drive
Los Angeles, California
(213) 663-8391

Henry Building
Upper Turin Road
Rome, New York
(315) 337-5600

European Office:
Building 64, Room 24
Schiphol Airport
Amsterdam, The Netherlands

AGE: 4 years - going on 5

EDUCATION: 679 years of college. 157 BA/BS, 43 MA/MS, 4 PhD degrees

DEPENDENTS: 300

MILITARY EXPERIENCE: On active service for USAF, USN, US Army, USMC

CLEARANCE: Top Secret

MARITAL STATUS: Publicly held subsidiary of Data Products Corporation

PROFESSIONAL EXPERIENCE: Informatics Inc.'s staff experience covers almost every conceivable type of computer system and application. A highly competent technical staff provides sophisticated, user-oriented systems analysis and programming services for a broad range of clientele.

SPECIAL QUALIFICATIONS: Of particular interest is our unusual competence in on-line and real-time systems. Over 80% of our work is in on-line applications. Here we offer exceedingly broad credentials in such applications as spacecraft checkout, missile range data handling, system monitor and control, and display/interrogation for military command and control.

OBJECTIVES: To be your software expert. To help you with programming and systems analysis. To supplement your staff. Or to tackle some of your tough, challenging problems.

LANGUAGE QUALIFICATIONS: Fluent in 250 Machine Languages -- including:

ABC/ACT/ALCOM/ALGO/ALGOL/ALGOL 60/ALGOL 3500/ ALTAC/AMCAL/AMICO/APT/ARGUS/ART/ART 418/ ASI & CSI/ATCOM/ATOLL/AUTOCODER-SPS/ H800 ARGUS/9AP/94AP/1401 AUTOCODER/1410 AUTO-CODER/7080 AUTOCODER/B-R 90/B-R 130/BAL/BAL-360/BALGOL/BE-FAP/BEEF/BEST/BIOR/BLIS/BLY-LANGUAGE/BOS-360-RFG/BPS-360/CDC924-924A SYMBOLIC/CEIR CODER/CGS/CL-1/COBOL/COBOL (AM AIR)/CODAP 1/ CODAP 2/CODOP/COMIT/COMPASS/ COMTRAN/COSEAL/CS-1/CS-2/DAP/DATA PROCESSOR/ 2 DIMENSIONAL/EASY/EASY CODER/EASY H400/ EASY II/ECAP/ECP-IL/ELAN/E Z CODE/FACT/FAF/ FAP/FARGO/FOR 130/FORTRAN/FORTRAN (NAA)/ FORTRAN II/FORTRANSIT/FRAP/FRUGAL/GAP/GAR-GOYLE/GECOM (GE)/GPSS/GPSS-II/GPX/G-15 MACH. LANG./IB-MAP/IB-MAP 7090-94/IBSFAP/ILLIAC/ INTERCOM/INTERCOM WOD/INTERCOM 500/INTERCOM 1000/IOCS 1401/IPL/IPL-V/IPS/ITERPRETIVE/ JOVIAL/JPL V/LAP/LINCOLN COMPILER/LISP/LO/ LO (CL-1)/LO (CL-11)/LUCID/MACRO/MAD/MAO/ MAP/MASS/MILITRAN/MOBIL/MOBL/MODSAP/MPL/ NCR 315 NEAT/NELIAC/NIKE ZEUS/NPL/OASIS/ OS-360/OSAP/OSAS/OSASA/PACT/PB250 SYMB/PL-1/ POP4 ASSEM/9 PAC/QD SURGE/RADCAP/RAWOOP/RCA-RPG/RCA-601 ASSEMBLY/ROAR/RPG/RTRIM/1401 RPG/ SAP/SAP-SPS/SATAN/SCAMP/SCAT/SCAT (PHOENIX)/ SCRIPT/SDS METASYMBOL/SDS-910-920-930-SYMBOL/ SE-9-AP/SEGAP/SIMCOM/SIMSCRIPT/SLAP/SLEUTH/ SLEUTH II/SLIP/SOAP II/SOAP IIA/SOAP (IBM650)/ SOL/SOS/SNAP/SNOBOL/SPAR/SPEEDCO/SPS-1/SPS-2/ SPS-1401/SPURT/STAR/STRAP/SURGE/SYMBOL/TAB-40/ TABSOL/TAC/TOPS/TRIM/TRIM III/TRIM-MW/ UII C-10/UII FLOWMATIC/UIII UTMOST/UMAP/UNI-CODE/UNIVAC II SALT/USE/USE COMPILER/USS SYMBOLIC/VAP/WIZ(GE)/X6/X60

AND, 120 Operating Systems Languages.

REFERENCES: Pacific Missile Range/Office of Naval Research/ Rome Air Development Center/Defense Communications Agency/Jet Propulsion Laboratory/National Military Command System Support Center/Space and Information Systems Division North American Aviation/IBM - Federal Systems Division/Radio Receptor Division General Instrument Corporation/ Radiation Inc./Remington Rand UNIVAC/Data Products Corporation/Perkin - Elmer/Space General/General Dynamics Electronics/Librascope/ Benson-Lehner/National Cash Register/Packard Bell/ Radio Corporation of America/General Electric - Syracuse/Astrodata/Thompson - Ramo - Wooldridge/ Western Union/Bureau of Standards/IBM/Douglas Aircraft